Human-like Action Segmentation for Option Learning

Jaeeun Shim and Andrea L. Thomaz

Abstract— **Robots learning interactively with a human partner has several open questions, one of which is increasing the efficiency of learning. One approach to this problem in the Reinforcement Learning domain is to use options, temporally extended actions, instead of primitive actions. In this paper, we aim to develop a robot system that can discriminate meaningful options from observations of human use of low-level primitive actions. Our approach is inspired by psychological findings about human action parsing, which posits that we attend to low-level statistical regularities to determine action boundary choices. We implement a human-like action segmentation system for automatic option discovery and evaluate our approach and show that option-based learning converges to the optimal solutions faster compared with primitive-action-based learning.**

I. Introduction

Our work is in the realm of social robotics. In particular, we focus on the problem of how robots can learn new tasks and skills from human demonstration. Considerable prior work on interactive robot learning from a human partner has been based on reinforcement learning (RL) [15].

The standard RL framework is a Markov decision process (MDP). Recent work has shown that temporally extended actions, known as options, can speed up learning and planning. Options are related to the actions of a semi-Markov decision process (SMDP) [13]. Thus, in Learning by Demonstration (LbD), agents can use the SMDP framework to make RL more efficient. To realize this goal, we need to ascertain how robots can automatically extract options from demonstrations. Robots could get option descriptions explicitly from a human, but our goal is for robots to automatically detect the high-level action boundaries of a task.

Inspired by mechanisms of human action perception, we propose a mechanism for robots to discover options. When humans parse actions, they certainly bring their experience to bear in a top-down manner. However, recent work shows that low-level features play an important role [4]. In their intentional action studies of infants, Baldwin et al. have shown evidence suggesting this role. In addition, their studies of adults suggest that humans can determine high-level action boundaries from the statistical regularities of low-level primitive actions.

In this work we show that robots can find options automatically from human-like action segmentation, and that these options enable them to more efficiently learn from demonstration. Our approach is as follows:

J. Shim is with the Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30308, USA jaeeun.shim@gatech.edu

A. Thomaz is with School of Interactive Computing, Georgia Institute of Technology, Atlanta, GA 30308, USA athomaz@cc.gatech.edu

This work is supported by NSF grant IIS-0960979.

1) **Human-like Action Segmentation:** We solve the option discovery problem inspired by human statistical learning from low-level primitive actions (detailed in Sec. III-A). The assumption is that it will be easier to predefine a low-level primitive action set than high-level actions or possible action sequences.
2) **Efficiency of the Learned Options:** Given the set of options learned in the previous step, we evaluate their effectiveness in speeding up the learning process. We show these options provide efficiency gains with respect to learning an optimal policy with just primitive actions. We also show that our option discovery method shows better performance than a state-of-the-art option discovery approach.

II. Related Work

There is much prior work in LbD, with examples of robots learning tasks from human partners with several different interaction techniques and learning methods (see [1] for a review). Several have used the RL framework for learning from human input [15]. [13] suggests that options can make RL more tractable, which is needed in LbD.

Once the option-based RL framework was proposed, many researchers focused on how to find options automatically. For example, [11] suggests that an agent learn options by finding subgoals or bottleneck states within the RL domain. In addition, several studies have described heuristics for options based on the frequency of visits during successful trajectories [10]. [16] and [6] also suggested algorithms of automatic option learning by finding subgoals in human trajectories. Most prior research on option learning has focused on regularities in the state space. Alternatively, our approach is based on regularities in the action space.

Option discovery is highly related to the action segmentation problem. In robot LbD, a common approach to segmenting actions is based on hidden markov models (HMM). For example, [5] discriminated hand-grasping gestures from continuous actions. Using HMMs, [7] and [2] also identified boundaries in action sequences. All of this work required a prior knowledge of primitive actions such as Kulic's technique [8]. After finding that primitive actions are constructed by sampling the frames of action sequences, [14] used a segmentation approach based on probabilistic correlation and the mimesis model. To achieve a more autonomous system, [9] developed robots that could learn primitive actions online during the demonstration. Using this approach, the robot does not need prior knowledge for action segmentation.

In previous work, primitive actions tend to be based on motions of the human body. For example, in [8] and [9],

primitive actions consist of the joint angle vectors of the human body. Alternatively, we focus on task-based primitive actions of object motion in a robot's workspace. In addition, unlike most systems in prior work, which are evaluated with expert users (usually the designers themselves), our system is evaluated with untrained everyday humans.

III. Background

A. Motivation from Human Psychology

People's natural action sequences are generally continuous. A person observing dynamic human action can find meaningful boundaries between smaller action chunks, i.e., they do action segmentation. Baldwin et al. researched the mechanisms by which humans parse intentional action and found evidence that points to this skill being the result of both top-down and bottom-up process [4], [3].

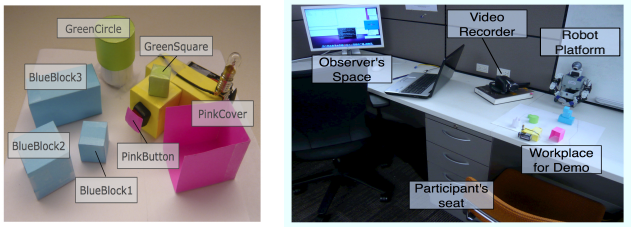
In one experiment, they showed that very young infants were able to determine the boundaries of intentional action (e.g., watching someone clean up in a kitchen) [4], which, given that the babies had no experience with kitchen cleaning, points to a bottom-up process for intentional action understanding based on low-level features.

In later work, they ran experiments with adults, testing a hypothesis that people attend to statistical regularities in low-level data, which are small-scale primitive actions, to determine action boundary choices. In [3], Baldwin et al. showed that statistical learning facilitates action segmentation. To show this, they took a set of random primitive actions and arbitrarily created higher-level "actions" from sets of low-level primitive actions. When they showed subjects a continuous sequence of primitive actions, the subjects were able to correctly identify the high-level action boundaries even though the actions had no higher-level semantic meaning. People used their experience with primitive actions to inform action boundary selection. If the probability of a transition between primitive actions is high, they are likely to be continuous high-level actions. On the other hand, if the probability of the transition from one primitive action to the next is relatively low, then it is likely to be an action boundary. Inspired by this human mechanism, our work proposes an LbD system that segments human dynamic actions in a similar fashion.

B. MDP and SMDP Preliminaries

RL enables an agent to learn an optimal policy of action. An agent can model the RL problem based on either an MDP or an SMDP [12], [13]. The MDP consists of a tuple $M =< S, A, P, R, \gamma >$, where S is a set of states, A is a set of actions, P is a transition function, R is a reward function, and γ is a discount factor. The transition function $P(s, a, s')$ represents the probability distribution of changing state s to s' over action a. The purpose of the MDP is to find a way of behaving, also known as policy, mapping states to actions, $\pi : S \times A \rightarrow [0, 1]$. The agent aims to identify a policy by maximizing the total reward received over time.

A more compact framework is the SMDP, which is defined over the MDP with a set of options O. An option $o \in O$, a generalization of actions that models a temporally extended action, is composed of three components: 1) an initial set of states S, 2) a termination set of states β, and 3) a policy π. In the SMDP, the agent finds policy $\mu : S \times O$ to determine the option that should be taken in any state.



(a) (b)

Fig. 1. (a) Domain Objects (b) Experimental Setting

Blue Objects	Green Objects	Pink Objects
BlueBlock1 Stacking(1)	GreenSquare Hanging(8)	PinkButton Pushing(15)
BlueBlock2 Stacking(2)	GreenCircle Hanging(9)	PinkSqureUncovering(14)
BlueBlock1 Unstacking(3)	GreenSquare Unhanging(10)	PinkSquare Covering(16)
BlueBlock2 Unstacking(4)	GreenCircle Unhanging(11)	PinkgSquare Moving(17)
BlueBlock1 Moving(5)	GreenSquare Moving(12)	
BlueBlock2 Moving(6)	GreenCircle Moving(13)	
BlueBlock3 Moving(7)		NonMotion(18)

TABLE I

Objects and Set of Primitive Actions (Action number)

IV. Implementation

A. System Domain

We conduct our experiments with seven colored objects distinguishable by the vision system of the robot based on color and size (Fig. 1(a)). From these objects, we define 17 different action primitives and 36 states.

Table I shows the set of primitive actions A, which are object-directed. For example, "Moving" actions are defined for all objects except the button object. "Stacking" actions correspond to the blue objects, and "Hanging/Unhanging" actions are conducted with the two green objects. "Covering/Uncovering" actions are done with the pink objects, and the pink button object has a "Pushing" action. The set of states, S, is defined by the relation among the objects. For example, three blue blocks can produce the four different relations as shown in Table II. Green and pink objects include three relations each. As a result, our experimental domain contains 36 discrete states (i.e., $4 \times 3 \times 3$).

In this work, we use an upper torso humanoid built from Bioloid kits and a webcam, shown in Fig. 1(b).

B. Detecting States and Primitive Actions

State and action detection consists of two modules: a perceptual phase and a detection phase. The perceptual phase receives the current vision data from the webcam. Our vision system uses OpenCV and tracks predefined objects with the blob-tracking algorithm based on colors and sizes. Each object blob has several properties including the location, orientation, size, and the number of blobs. The current vision

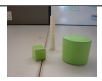
Objects	Possible States			
Blue				
Green				
Pink				

TABLE II

POSSIBLE STATES WITH EACH OBJECTS

input is compared with the previous vision input to determine whether any objects have changed between the current and previous frame. If so, the system goes to the detection phase, in which the system identifies which primitive action and state changes occurred. To determine this, the system observes the properties of each blob and detects which blobs have changed. Then, based on predefined property change expectations, the system can determine primitive actions and states. The action set also includes "NonMotion," out-of-vocabulary primitive actions are classified as "NonMotion."

C. Human-Like Action Segmentation to Find Options

In our system, the robot identifies options using the human-like action segmentation mechanism. During the human demonstration, the robot generates a statistical model of the transition probabilities T_{ij} between the different primitive actions detected. T_{ij} indicates the probability of primitive action j occurring after i. The robot uses this to discriminate action boundaries using statistical regularities, as described in Section III-A. Our system determines action boundaries by identifying action transitions with lower probabilities than $threshold$, which is learned from the transitions seen across all users. For each primitive action i, the system finds the probability of the most frequently occurring transition, $MaxProb_i$. The robot determines the $threshold$ by:

$$threshold = \underset{i \in PrimitiveActions}{\text{average}} \; MaxProb_i$$

With this, the robot discriminates action boundaries from a sequence using the following:

$$isBoundary(i,j) = \begin{cases} true & \text{if } T_{ij} < threshold \\ false & \text{otherwise} \end{cases}$$

The probability of a transition that is higher than the $threshold$ indicates that the transition of these primitive actions can represent a meaningful high-level action sequence.

With this action segmentation, the robot automatically detects options, which are defined as a meaningful set of primitive actions in a sequence. In other words, if a set of primitive actions $\{a_1, a_2, a_3\}$ is determined to be a high-level action, then an option is defined with the following three components: 1) initial state s, the beginning state of primitive action a_1, 2) termination state s', the end state of primitive action a_3, and 3) policy π, defined by a set of primitive actions $\{a_1, a_2, a_3\}$.

V. EVALUATION

Our evaluation consists of two parts: the action-segmentation session and the learning session. In the action-segmentation session, the robot observes the demonstration of real human subjects and discriminates high-level actions to find options. In the learning session, we evaluate whether the detected options are efficient for robot LbD.

A. Action Segmentation Session

In this experiment, participants sat in front of the robot and had a bounded workplace with the objects to be used for the demonstration. The experimenter and a video recorder were situated next to the robot and the participant (Fig. 1(b)).

We recruited 18 participants (8 female) from the campus community for this action segmentation session (i.e., the training session). When the experiment began, we first explained the 17 primitive actions (Table I). For each participant, we explained these primitive actions in a different random order to control for any bias the instructions created on one's subsequent use of the primitive actions. Importantly, we talked about only primitive actions and did not give any information about high-level actions of the domain. After the participants had learned and memorized the primitive actions sufficiently, we gave instructions about the demonstration session. Participants then demonstrated the primitive actions in any order they chose for twenty minutes.

In a separate experiment (i.e., the test session), we recruited two more subjects and gathered a test dataset. Here, we did not mention the primitive actions to the participants but simply mentioned the name of the three high-level actions that could be performed with the object set: Stack, Exchange, and Button (detailed in Table III). We then requested that the participants demonstrate these actions as they imagined, performing each action twice in any order they chose.

B. Action Segmentation Results

In the experiment, each participant demonstrated an average of 210 primitive actions during the twenty minutes (Max: 262; Min: 160; SD: 34.75). The aggregate set of all demonstrations included 3,772 primitive actions observed over 360 minutes. The observed data also include noise, since our vision system is not entirely reliable. The system builds an action transition model (Sec. IV-C) from each of the individual datasets and for the aggregate dataset, a total of 19 models with the unfiltered noisy data.

Fig. 2 shows the results of action segmentation on the test data, using these models. Fig. 2(a), is the data from test subject 1, and Fig. 2(b) is the data from test subject 2. The sequence of numbers in each row indicates the sequence of recognized primitive actions demonstrated by each of the two participants during the test session. These numbers correspond to primitive actions shown in Table I. Given a sequence of primitive actions, the robot inferred the action boundaries based on each of the 19 trained models. The top five rows show the action segmentation results using five different individual models. (We show only 5 out of the 18 due to limited space). The sixth row shows the action

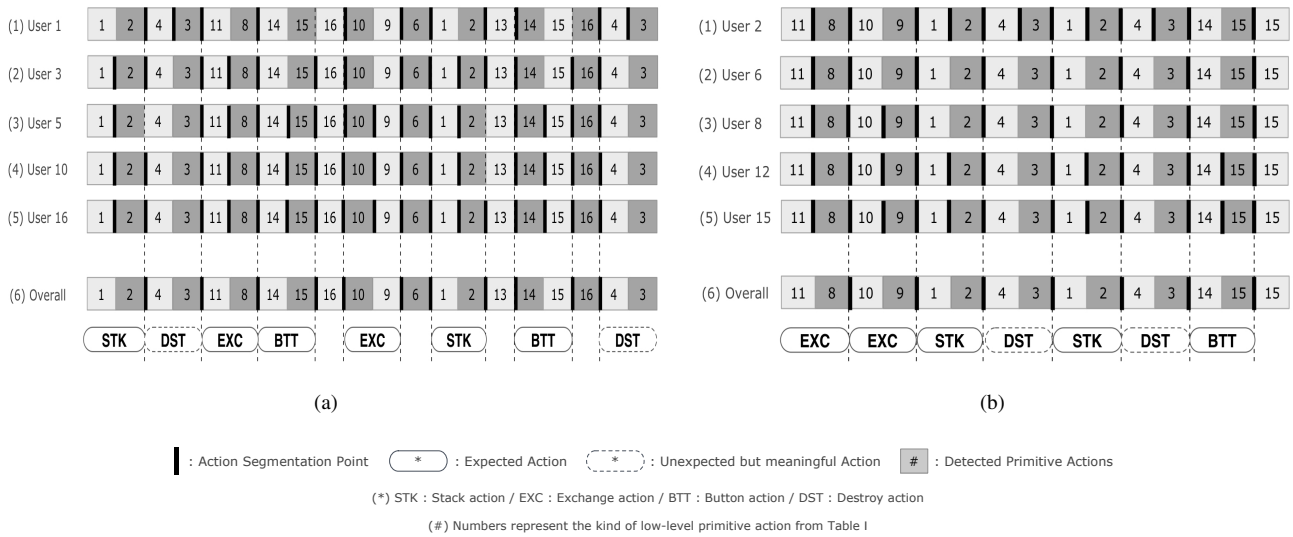


Fig. 2. Action segmentation results in (a) test data 1 (b) test data 2: One can see that the results from individual data sets (1-5) cannot discriminate every action boundary whereas the results from the overall dataset (6) shows that all actions are segmented accurately in both test cases.

High-level action name	Expected sequence of primitive actions
"Stack"	stacking(BlueBlock1) - stacking(BlueBlock2)
"Exchange"	Unhanging(GreenSquare) - Hanging(GreenCircle) Unhanging(GreenCircle) - Hanging(GreenSquare)
"Button"	Uncovering(PinkSquare) - Pushing(PinkButton)

TABLE III
EXPECTED OPTIONS

segmentation results using the aggregate model from the entire training data.

In the experiment, test subject 1 demonstrated their six actions in the following order: Stack-Exchange-Button-Exchange-Stack-Button. The robot observed this action sequence as the combination of primitive actions in Fig. 2(a). The action segmentation results from the 18 different individual data indicate that the individual models were unable to identify every expected action boundary, as shown in Table III. For example, (2)-(5) in Fig. 2(a) show that the statistical models put an action boundary between "Stack blue block 1" and "Stack blue block 2" primitive actions, which were expected to be detected as the "Stack" high-level action. Many of them could not discriminate between "Exchange" and "Button" high-level actions (e.g., (3)-(5) in Fig. 2(a)). However, the results from the aggregate dataset discriminates high-level actions exactly as we expected. The segmented actions with the overall datasets match Table III (row (6) in Fig. 2(a)).

Subject 2 performed actions in the following order: Exchange-Exchange-Stack-Stack-Button-Button. Again, we observed that the statistical models from the individual training data could not discriminate the high-level actions entirely. However, with the overall data, the robot found the expected high-level action boundaries (Figure 2(b)).

Thus, since the agent can successfully discriminate meaningful high-level actions from the primitive actions, we propose that these high-level actions can be used as options in an SMDP framework.

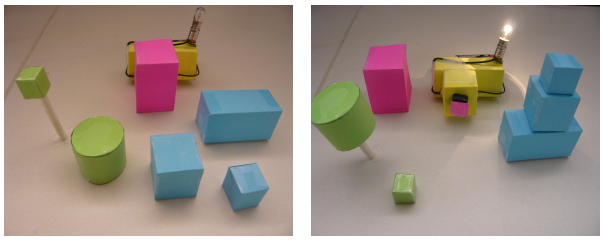
C. Learning Session

In this part, our system evaluates the utility of the options found in the segmentation process. These learning evaluations are run in simulation only. To evaluate the learning efficiency of our options, we build the MDP with the primitive actions and the SMDP with the options from the human-like action segmentation system.

For the MDP, state set S and action set A are defined as we explained in Section IV-A. Based on the overall data of the demonstration during the experiment, the robot builds a statistical model of transition function $P(s, a, s')$ among the detected states s, s' and primitive action a. A reward function R is defined by the goal state. If final state s' of $P(s, a, s')$ is the goal state, we assign 20 as the value of $R(s')$. Otherwise, the reward $R(s)$ is -1.

The robot also generates the SMDP framework, which is defined over the MDP with the set of options O. Options are the resulting set from the human-like action segmentation mechanism. Our system discovers 6 options from "Stack," 6 options from "Destroy," 12 options from "Exchange," and 12 options from "Button" actions. As a result, from the 36^{18} possible transitions in the initial MDP framework, our system discovers 36 meaningful options.

For both the MDP and the SMDP, we can determine the learning efficiency by observing how fast the robot finds the optimal policy from an initial state to the goal state. For the learning session, initial states are selected randomly in each episode. The goal state is arbitrarily fixed as the state shown in Fig. 3(b). In this evaluation, the robot learns policies with the Q-learning algorithm [12]. As the number of episodes increases, the policy converges to the optimal policy. Furthermore, Q-learning guarantees that the



(a) (b)

Fig. 3. (a) Initial state (b) Goal state

optimal policy will be found. Therefore, we use the rate of convergence to the optimal policy as an indication of learning efficiency.

Some prior research has also proposed the automatic option learning algorithms. To claim the contribution of our approach, we need to show that our option learning algorithm leads to the better performance than other computational option learning algorithms. Thus, we also compare the performance of our proposed method with the automated method. From the various prior work, we select and simulate Stolle and Precup's option learning method [11]. In this algorithm, an agent extracts options by computing the most-often-visited states, namely bottleneck states, when the agent repeats the relevant tasks. From the bottleneck states, the agent specifies an initiation set and a termination condition, and then the agent also learns the internal policies using standard RL algorithms.

To simulate this algorithm in our specific domain, we select the number of start states s_i randomly and fix the target states s_T as the state shown in Fig.3(b). For each pair $< s_i, s_T >$, the agent performs the Q-learning algorithm N times, 1,000 in our system. Meanwhile, for all states, the agent counts the total number of times $n(s)$ that each state is visited. Then, we can pick the bottleneck state by finding $s_b = argmax_s n(s)$. After detecting the bottleneck state, the agent computes the threshold, which is $\overline{n}(s, s_b) = avg_s n(s, s_b)$ where $n(s, s_b)$ is the number of times each state s occurs on paths that go through s_b. From this threshold, we can determine the initiation set for the option by identifying all the states s for which $n(s, s_b) > \overline{n}(s_b)$ and then determine each internal policy using the Q-learning. To discover the same number of options as we found in our proposed approach, the agent repeats this algorithm until 36 options are reached.

For all frameworks (i.e., the MDP with primitive actions, the SMDP with our options, and the SMDP with bottleneck-based options), the robot runs the learning algorithm separately and finds the optional policy. After finishing each episode, the robot finds the set of behaviors from state 1 (Fig. 3(a)) to the target goal state (Fig. 3(b)) and calculates the accumulated reward.

D. Learning Results

To evaluate the performance of our algorithm, we compared the convergence rates of Q-learning based on our

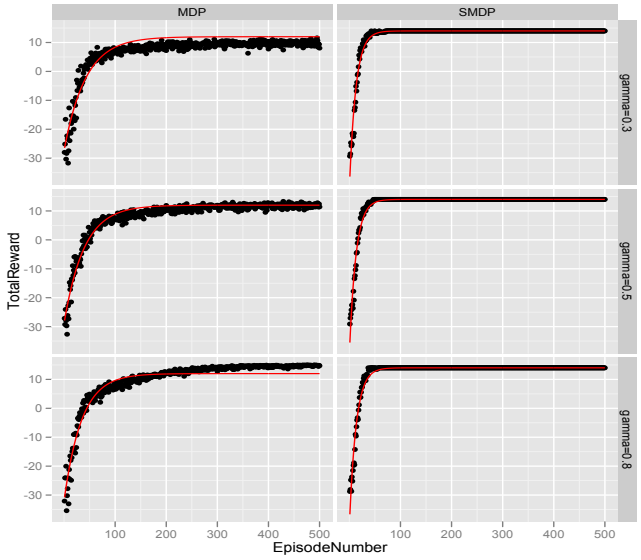


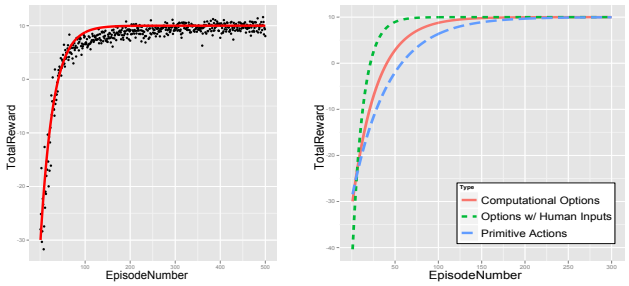
Fig. 4. Simulation Result1: 1) total rewards of planning to the goal state with Q-learning (black points) and 2) estimated graph (red lines) ($\gamma = 0.3$)

	MDP	SMDP
$\gamma = 0.3$	42.0168 (c=0.0238)	12.6743 (c=0.0789)
$\gamma = 0.5$	38.9105 (c=0.0257)	12.7877 (c=0.0782)
$\gamma = 0.8$	34.2466 (c=0.0292)	13.2275 (c=0.0756)
Average	38.3913	12.8964

TABLE IV
CONVERGENCE RATE

options (SMDP) with primitive actions (MDP) and also with bottleneck-based options.

1) Comparison with Primitive Actions: Figure 4 depicts the simulation results of convergence with the SMDP and the MDP. The x-axis indicates the number of episodes performed in Q-learning, and the y-axis shows the total reward gained during planning from the initial to the goal state. The robot plans a way of behaving in each state based on the policies found from the Q-learning algorithm. Fig. 4 shows that both Q-learning results, based on the MDP and the SMDP, converge to the optimal policy. By comparing the graphs in the MDP with those in the SMDP, we can observe that the SMDP-based learning agent converges to the optimal policy faster than the MDP-based learning agent. To evaluate the convergence rates clearly, we estimate each total-reward distribution with the exponential graph, which is over-laid as the red line in each figure. We formulate each graph with the following equation, $y = -a + b\exp(-c)$, by observing all plots. For the estimation, we use the non-linear least-squared regression method. In the exponential distribution, the convergence rate depends on the exponentiation, parameter c. Simply, $1/c$ can determine the rate of convergence. The smaller value of the convergence rate indicates a faster convergence speed. Table IV shows parameter c and its corresponding convergence rates. SMDP-based learning clearly shows a smaller average convergence rate than MDP-based



(a) (b)

Fig. 5. Simulation Result2: (a) total rewards of Q-learning planning to the goal state with bottleneck-based options; (b) a comparison of the performance of MDP with primitive actions, SMDP with computational options, and SMDP with our options ($\gamma = 0.3$)

learning. This result indicates that the system shows better learning efficiency with the human-derived options than with primitive actions. Therefore, our proposed way of detecting the options with the human-like action segmentation system is a viable way to learn options automatically.

2) Comparison with Other Automated Options: We compare the performance of our proposed method with the automated method. Fig. 5(a) illustrates the result of convergence using the bottleneck options. As shown in this figure, this algorithm also converges to the optimal policy. To compare the convergence rates, we calculate the estimated exponential graph, depicted in Fig. 5(b). This figure shows that our approach enables an agent to converge to the optimal policy faster than the other two methods. The bottleneck options from [11] converge to the optimal policy faster than the primitive-action-based algorithm, but they are slower than our human-derived options. Therefore, we can show that our approach can increase learning efficiency compared with not only the primitive-action-based MDP but also automated-option-based SMDP.

VI. Conclusion and Future Work

This work is inspired by psychological findings related to human action parsing [3]. Based on this work, we implemented an LbD system that builds a statistical model of demonstrated primitive actions and then uses this to infer high-level actions for the domain. We show that these high-level actions can be used as temporally extended actions in an RL framework, i.e., options. To evaluate the system, we compared the learning efficiency of general learning framework with primitive actions (MDP) with that of our learning framework with options (SMDP). This system, trained with data from naïve human subjects, finds that the robot cannot discriminate every expected high-level action with the individual training data, but can find all expected high-level actions successfully when using the aggregate data across all 18 people.

In the learning session, we showed that the learning efficiency with the options that the robot extracts using the human-like action segmentation system is higher than when using just the primitive actions. We also show that our human-derived options are more efficient than those found with a typical automatic option algorithm based on bottleneck states. Thus, we conclude that a robot can automatically find meaningful options from a human-like action segmentation system if the statistical models are learned in an aggregate fashion with a variety of human subjects.

References

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robot. Auton. Syst.*, 57(5):469–483, 2009.

[2] T. Asfour, F. Gyarfas, P. Azad, and R. Dillmann. Imitation learning of dual-arm manipulation tasks in humanoid robots. In *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 40 –47, dec. 2006.

[3] D. Baldwin, A. Andersson, J. Saffran, and M. Meyer. Segmenting dynamic human action via statistical structure. *Cognition*, 106(3):1382–1407, March 2008.

[4] D. A. Baldwin and J. A. Baird. Discerning intentions in dynamic human action. *Trends in Cognitive Sciences*, 5(4):171–178, 2001.

[5] K. Bernardin, K. Ogawara, K. Ikeuchi, and R. Dillmann. A sensor fusion approach for recognizing continuous human grasping sequences using hidden markov models. *Robotics, IEEE Transactions on*, 21(1):47 – 57, feb. 2005.

[6] G. Comanici and D. Precup. Optimal policy switching algorithms for reinforcement learning. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1*, AAMAS '10, pages 709–714, Richland, SC, 2010. International Foundation for Autonomous Agents and Multiagent Systems.

[7] T. Inamura, I. Toshima, and H. Tanie. Embodied symbol emergence based on mimesis theory. *International Journal of Robotics Research*, 23:4–5, 2004.

[8] D. Kulić, W. Takano, and Y. Nakamura. Incremental learning, clustering and hierarchy formation of whole body motion patterns using adaptive hidden markov chains. *Int. J. Rob. Res.*, 27(7):761–784, 2008.

[9] D. Kulić, W. Takano, and Y. Nakamura. Online segmentation and clustering from continuous observation of whole body motions. *Trans. Rob.*, 25(5):1158–1166, 2009.

[10] A. McGovern and A. G. Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 361–368, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[11] M. Stolle and D. Precup. Learning options in reinforcement learning. In *Proceedings of the 5th International Symposium on Abstraction, Reformulation and Approximation*, pages 212–223, London, UK, 2002. Springer-Verlag.

[12] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[13] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: a framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, 112:181–211, August 1999.

[14] W. Takano and Y. Nakamura. Humanoid robot's autonomous acquisition of proto-symbols through motion segmentation. In *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 425 –431, dec. 2006.

[15] A. L. Thomaz and C. Breazeal. Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artif. Intell.*, 172:716–737, April 2008.

[16] P. Zang, P. Zhou, D. Minnen, and C. Isbell. Discovering options from example trajectories. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1217–1224, New York, NY, USA, 2009. ACM.